IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received February 10, 2020, accepted March 9, 2020, date of publication March 17, 2020, date of current version March 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2981395

Design Automation of Approximate Circuits With Runtime Reconfigurable Accuracy

GEORGIOS ZERVAKIS, HUSSAM AMROUCH, (Member, IEEE), AND JÖRG HENKEL, (Fellow, IEEE)
Chair for Embedded Systems, Department of Computer Science, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany

Corresponding author: Georgios Zervakis (georgios.zervakis@kit.edu)

This work was supported in part by the German Research Foundation (DFG) Project "ACCROSS".

ABSTRACT Leveraging the inherent error tolerance of a vast number of application domains that are rapidly growing, approximate computing arises as a design alternative to improve the efficiency of our computing systems by trading accuracy for energy savings. However, the requirement for computational accuracy is not fixed. Controlling the applied level of approximation *dynamically* at runtime is a key to effectively optimize energy, while still containing and bounding the induced errors at runtime. In this paper, we propose and implement an automatic and circuit independent design framework that generates approximate circuits with dynamically reconfigurable accuracy at runtime. The generated circuits feature varying accuracy levels, supporting also accurate execution. Extensive experimental evaluation, using industry strength flow and circuits, demonstrates that our generated approximate circuits improve the energy by up to 41% for 2% error bound and by 17.5% on average under a pessimistic scenario that assumes full accuracy requirement in the 33% of the runtime. To demonstrate further the efficiency of our framework, we considered two state-of-the-art technology libraries which are a 7nm conventional FinFET and an emerging technology that boosts performance at a high cost of increased dynamic power.

INDEX TERMS Approximate computing, approximate design automation, dynamically reconfigurable accuracy, low power.

I. INTRODUCTION

Recent research by Intel, IBM and Microsoft has demonstrated that there is a large number of application domains, e.g., machine learning, that exhibit an intrinsic error tolerance [1]. It is shown that in such applications the 70%, on average, of their energy consumption is dissipated in computations that can be approximated [2], if a certain accuracy level is ensured. Approximate computing, exploits the inherent error resilience to trade accuracy for gains in other metrics (e.g., energy, performance etc). Driven by this high potential for energy reduction, designing approximate circuits gains significant research interest. Despite the fact that such applications can tolerate errors in the performed computations, an application's requirements for computational accuracy is not fixed [3] and it may vary over time and with respect to the application's state. Therefore, error resilient applications necessitate to switch between different levels of accuracy at runtime. However, approximate circuits traditionally apply fixed approximation, e.g., [4]–[41], neglecting the potential for dynamic accuracy reconfiguration. In this work, we respond to this necessity and address for the first time the problem of generating approximate circuits that can switch between varying accuracy levels at runtime and also support fully accurate execution.

Hardware-level approximation mainly targets arithmetic units as they constitute the key components of a vast number of error tolerant applications, e.g., signal processing, image processing, and neural networks [2]. In [4]–[7], [8]–[14], and [15]–[20] operation specific approximation techniques are proposed to build approximate adders, multipliers and dividers, respectively. In [21]–[34] approximate design automation frameworks are proposed to ease the generation of approximate circuits. In [35]–[39] complex application specific approximate accelerators are designed. In [35], [36] approximate implementations of the coordinate rotation digital computer (CORDIC) design are proposed, while in [37], [38] approximate Fast Fourier Transformation (FFT) circuits are generated. In [39] approximate Sum of Absolute Differences (SAD) kernels are proposed to accelerate

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

an H.265/HEVC encoder. In [40], [41] machine learning is employed to build large approximate accelerators out of approximate libraries containing simple approximate circuits. However, all the aforementioned works apply fixed approximation at design time that cannot be modified at runtime. The ever-growing need for controlling the accuracy of the approximations at runtime is highlighted in [42]–[47] where approximate circuits that exhibit dynamic accuracy reconfiguration are proposed. However, all these works are operation specific and they induce significant overheads at the accurate operation. In addition, they mainly support only one approximation level [42]–[45] and thus, they cannot adapt the approximation level to the application's needs.

In this paper, we are the first to propose and implement an automatic design framework for generating approximate circuits that feature dynamically reconfigurable accuracy and *support several accuracy levels*. The proposed framework replaces the circuit's wires with "approximate switches" and reduces the circuit's power consumption by systematically constraining its switching activity. Given the desired accuracy levels, the proposed framework identifies which wires have to be approximated at each level. At runtime, the selection of the desired accuracy level is performed using a control signal that activates the corresponding approximate switches. To identify the approximated wires, we employ a power-aware heuristic optimization procedure. Our framework is circuit agnostic and can be applied to any combinational circuit, seamlessly extending any hardware design flow.

Extensive experimental evaluation demonstrates the efficiency of the approximate circuits generated by the proposed framework. Compared to the accurate circuits, the ones generated by our framework feature similar performance while reduce the energy consumption by 22% and 29%, on average, when operating at the accuracy levels with 1% and 2% error bounds respectively. Our proposed and implemented framework is evaluated using various circuits from the commercial Synopsys tool flows at the state-of-the-art 7nm FinFET technology node. Furthermore, we additionally evaluate our framework at the 7nm Negative Capacitance FET (NCFET) technology demonstrating the effectiveness of our framework in such a promising new technology that is very rapidly emerging for future computing. Compared to conventional CMOS technology, at the same operating voltage, NCFET achieves much higher performance at the cost, however, of much higher dynamic power consumption. Our analysis demonstrates how our proposed reconfigurable approximation enables the exploitation of NCFET technology by eliminating the energy/power downside of the NCFET circuits. Applying our framework to NCFET-based circuits retains their high performance and improves their efficiency (represented by delay-energy product) by up to 1.7×. Finally, we also demonstrate the efficiency of our framework when targeting fixed approximation as well as at application level considering approximate Neural Network inference.

Our novel contributions in this work are as follows:

- We propose and implement, for the first time, an automated and circuit-independent design framework that produces dynamically reconfigurable approximate circuits that support several accuracy levels at runtime.
- We employ a power-aware heuristic approach that efficiently identifies the wires that will be approximated. Our approach avoids performing time consuming circuit simulations and evaluates at high level the power and error values of the approximated circuit.
- Our framework is evaluated at 7nm FinFET as well as emerging NCFET technologies. This is the first time that approximate computing is employed and evaluated to address the energy downside of the NCFET technology.

The rest of the paper is organized as follows: Section II discusses the generation of dynamically reconfigurable approximate circuits and Section III describes our framework. In Section IV the proposed framework is experimentally evaluated and Section V discusses the prior art in the field of approximate design. Finally, Section VI concludes this work.

II. APPROXIMATE CIRCUITS WITH RECONFIGURABLE ACCURACY

In this section the proposed method for generating approximate circuits with reconfigurable accuracy is described. The vast majority of the existing approximate circuits feature only a single accuracy level. However, at application level, the requirement for numerical accuracy is not static and the impact of approximation, to the end quality, highly depends on the inputs characteristics [3]. Moreover, an approximation that is optimal for an input distribution in terms of energy efficiency and/or induced error, it might be sub-optimal for a different one. In addition, it is highly possible that the quality requirements of an application might vary over time. For example, in the case of an embedded system, as its battery level goes down, in some application domains (e.g., image processing), higher error values might be accepted in order to prolong its life. To address the aforementioned inefficiencies, we propose a framework for generating approximate circuits that exhibit dynamically reconfigurable accuracy. In the generated circuits, the desired accuracy level can be dynamically selected at runtime by setting accordingly the value of a control signal. Selecting the accuracy level is performed seamlessly and the generated circuits can switch on the fly among the accuracy levels without any latency overhead.

The main design target of the approximate circuits is to minimize the power/energy consumption. A circuit's power dissipation is estimated by [48]:

$$P \approx P_{static} + P_{dynamic}, \tag{1}$$

where $P_{dynamic}$ is given by [48]:

$$P_{dynamic} = aCV^2f. \tag{2}$$

The parameter a is defined by the circuit's switching activity, C is the total circuit's capacitance, and V and f are the

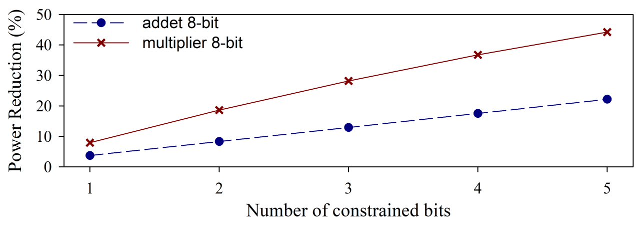


FIGURE 1. Examining the impact of limiting the switching activity on the power consumption. An 8-bit adder and an 8-bit multiplier are examined. In this example, the switching activity is constrained by keeping some of the inputs' least significant bits constant. The circuits are obtained from the Synopsys DesignWare Library (M-2016.12), they are optimized for speed, and the implementation of both circuits is pparch.

operating voltage and frequency values respectively. Therefore, given (1) and (2), the power consumption highly depends on the circuit's area, voltage value, and switching activity. Hardware approximation techniques produce, mainly, simpler circuits (e.g., logic simplification [5]) to reduce the circuit's area and/or decrease the operating voltage value below its nominal value (e.g., Voltage Overscaling [23]). Simplifying the circuit can also result to lower delay, whereas reducing V decreases only the power consumption. In this work, we systematically limit the circuit's switching activity to reduce its dynamic power and eventually its total power consumption. To elucidate the impact of limiting the switching activity on the circuit's power, a representative example is illustrated in Fig. 1. In this example, we constrain the switching activity of two different arithmetic circuits (i.e., 8-bit adder and multiplier) by keeping the least significant bits of the circuit's inputs constant. For each circuit, 10^5 random generated inputs are used. On average, the power consumption of the 8-bit adder and multiplier decreases by 13% and 27%, respectively. The power reduction of the adder ranges from 3.6% up to 22.2%, while the respective values for the multiplier are 7.9% and 44%. Although the power consumption of both circuits decreases significantly, the multiplier features higher power savings. The 8-bit multiplier is considerably more complex circuit than the 8-bit adder and thus, its power consumption is benefited more by decreasing a. However, in Fig. 1, the average error value[1] of the adder ranges from 0.7% up to 20%, while the error of the multiplier ranges from 3% up to 49%. Thus, despite the high power reduction attained, the output error value is also very large. As a result, directed approximations are required to limit the switching activity and maximize the power savings while constraining the error value.

One of the most effective approximation techniques to generate approximate circuits is the netlist approximation [21]–[24]. A representative example of netlist approximation is the netlist pruning [21], [24]. Given an optimal accurate implementation, netlist pruning replaces the circuit wires by constant values. Hence, by removing any floating cells, a less complex circuit is obtained in which less power

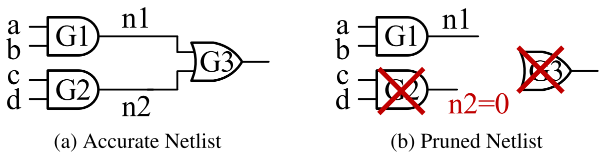


FIGURE 2. The approximate netlist (b) produced by applying netlist pruning with n2=0 on the accurate netlist of (a).

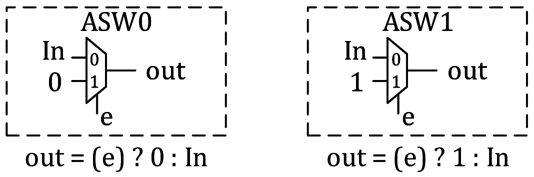


FIGURE 3. The approximate switches ASW0 and ASW1 used in our framework to produce approximate circuits with reconfigurable accuracy.

is consumed but also errors are produced due to the wire-by-constant substitution. Netlist pruning offers a very fine grained control of the performed approximations and thus, very efficient approximate circuits are obtained. Fig. 2 depicts an example of the netlist pruning technique. Assuming a uniform input distribution, the output of an AND gate is equal to logic '0' for the 75% of the outputs. Hence, the wire $n2$ (output of the AND gate $G2$) is replaced by a constant '0' to produce the pruned circuit. The resulting approximate circuit, depicted in Fig. 2b, equals to one AND gate (i.e., 2 gates saving) and features an error rate of 18.75% (3/16).

Despite its high efficiency, netlist pruning leads to very different circuit implementations and prohibits the runtime accuracy reconfiguration. In this paper, we exploit the efficiency and the fine grained control delivered by netlist-specific approximations and enable approximate circuits with dynamically reconfigurable accuracy. We decrease the circuit's switching activity by dynamically limiting the toggling of a set of circuit wires. Similar to the netlist pruning, we identify the wires that have to be approximated as well as the respective value to be used for the approximation ('0' or '1'). Then, the identified wires are replaced by a switch that selects between the wire's value and the extracted approximate one. We use the two approximate switches, named ASW0 and ASW1, depicted in Fig. 3. When the switch ASW0 is off (enable signal is 0), it outputs the wire's value, i.e., accurate mode. When ASW0 is on (enable signal is 1), it outputs a constant '0', i.e., approximate mode. Similarly, ASW1 produces a logic '1' when it is on. Hence, by tuning off/on the switches, we can dynamically switch to accurate/approximate execution. At the approximate mode, the output of the switches is constant, the circuit's switching activity is reduced and thus, its power consumption decreases.

The accurate circuit of Fig. 2a is reused to give an example of the proposed netlist modification. Replacing the wire $n2$ with ASW0 produces the approximate circuit of Fig. 4. When the enable signal $ctrl$ is '0', the value of the wire $n2'$ equals to the value of the wire $n2$ and accurate results are obtained. When $ctrl$ is set to '1', $n2'$ equals to '0' and, similarly to Fig. 2b, the error rate is 18.75%. However, the proposed

[1]The Relative Error Distance (RED) is used as an error metric: $RED = |\frac{accurate-approximate}{accurate}|$ [14].

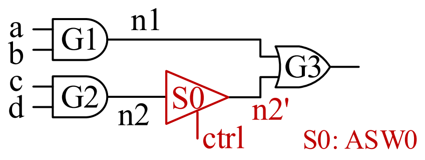


FIGURE 4. Example of using ASW0 instead of applying netlist pruning (see example of Fig. 2). When ctrl=0 accurate computations are performed. When ctrl=1 approximate results are obtained with error rate 18.75%.

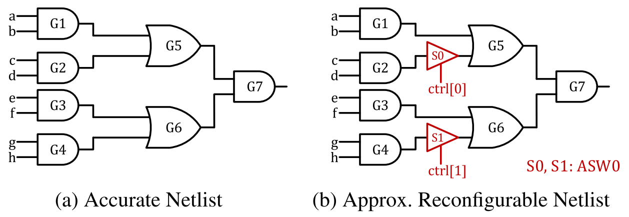


FIGURE 5. Example of enabling varying accuracy levels using the “approximate switches”. When the control signal ctrl=2’b00 fully accurate operation is achieved. When ctrl=2’b01 or ctrl=2’b11 approximate execution is enabled and the error rate is 8.2% and 12.9%, respectively.

method is not limited to only accurate and approximate modes. It can be easily extended to support varying accuracy levels. By identifying which wires have to be approximated in each accuracy level, a multi-bit control signal can be used to dynamically enable only the respective switches that deliver the desired accuracy. Fig. 5 illustrates a simple example of how the required accuracy level can be dynamically selected. Fig. 5b is an approximate counterpart of Fig. 5a that has two switches (*S0* and *S1*) and a 2-bit control signal *ctrl[1:0]*. If *ctrl*=2’b00, all the switches are off and accurate results are obtained. If *ctrl*=2’b01, the switch *S0* is on, the switch *S1* is off, and the circuit’s error rate is 8.2%. Finally, if *ctrl*=2’b11, both switches are on and the error rate becomes 12.9%. When replacing a wire with an approximate switch, when the switch is on, the output of the switch is constant and thus, the switching activity of the circuitry driven by this wire decreases. For example, in Fig. 5b, the gates *G5*, *G6*, and *G7* are affected by the approximated wires. The more gates are benefited by neglecting the switching activity of the approximated wire, the higher power reduction is achieved.

In Fig. 5b we showed how we modify the circuit’s netlist and achieve dynamic accuracy reconfiguration. Modifying accordingly any circuit’s netlist while satisfying given accuracy constraints can become a very complex task, especially as the circuit’s complexity increases. Moreover, compared to the accurate circuit, the power decreases when operating at an approximate level since the switching activity is constrained. The added circuitry will consume static power in both the accurate and approximate operation. Nevertheless, a high reduction of the switching activity will lead to high dynamic power reduction. Considering that in the state-of-the-art FinFET technology the dynamic power dominates the static one [49], a high decrease of the dynamic power will compensate the static power of the approximate switches and eventually the total power consumption will decrease significantly. However, due to the added logic gates, the power increases when operating at the accurate level. Therefore, a systematic approach is required to enable the generation of such approximate circuits with dynamically reconfigurable accuracy and maximize the gains while minimizing the overheads. To this end, we propose the RETSINA (appRoximatE circuiTS wIth recoNfigurable Accuracy) framework. RETSINA is an automated logic synthesis framework, written in C and Python, that can seamlessly extend any typical design flow to generate approximate circuits with reconfigurable accuracy at runtime. The proposed framework operates over the gate-level netlist and thus, it can be applied to any circuit. In this work, we limit our analysis to combinational circuits only.

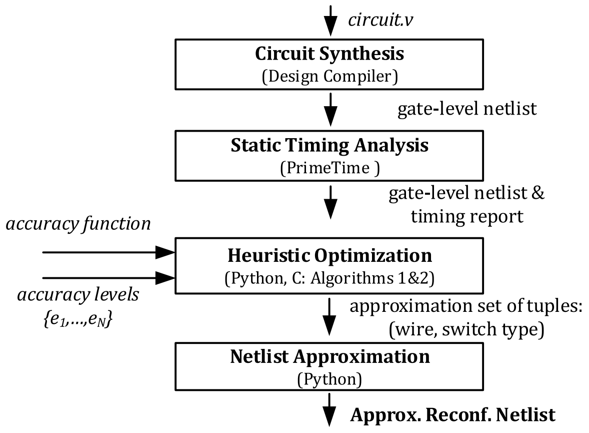


FIGURE 6. Abstract overview of the flow used by the proposed framework to generate approximate circuits with reconfigurable accuracy.

III. THE PROPOSED RETSINA FRAMEWORK

In this section, we discuss the optimization problem of producing the previously described approximate circuits that feature dynamically reconfigurable accuracy at runtime. Then, the proposed RETSINA framework is described. An abstract overview of the proposed framework is illustrated in Fig 6. Briefly, RETSINA operates as follows: given the hardware description of a circuit, RETSINA synthesizes the circuit and extracts the accurate gate-level netlist. Then, it performs a Static Timing Analysis (STA) and extracts the circuit’s timing information. Next, given i) a user provided quality function, ii) the desired accuracy levels, and iii) the previously obtained netlist and timing report, RETSINA applies a heuristic optimization to identify the netlist’s wires that will be approximated as well as the respective switch type (ASW0 or ASW1). Finally, the accurate netlist is approximated by replacing the extracted wires with the respective switches. During the netlist approximation, an input control signal is also added to the netlist. For N accuracy levels, the width of the input control signal is N-1 bits. The enable signal of each approximate switch is connected to a bit of the input control signal. Hence, at runtime, the user can switch between the accuracy levels by setting accordingly the input control signal. After the netlist modification Design Compiler is used to fix any fanout constraints.

The generated approximate circuits feature varying accuracy levels that can be dynamically selected using the control signal. Assume N accuracy levels with error bounds $\{e_1, \ldots, e_N\}$. For each level i, RETSINA has to identify the approximation set $\mathbf{X_i}$ that minimizes the power consumption and satisfies the error constraint (e_i) when the approximate circuit is operated at this level i. The set $\mathbf{X_i}$ comprises the circuit wires that have to be approximated as well as the respective approximate switch type. The union $\mathbf{Y_N} = \bigcup_{i=1}^{N} \mathbf{X_i}$ constitutes the final approximate circuit. The latter is generated by replacing in the accurate netlist the wires of $\mathbf{Y_N}$ with the respective switches. A different enable signal is used to activate the switches of each set $\mathbf{X_i}$ (i.e., dynamically select the accuracy level). For the sake of simplicity we assume that the accuracy levels are sorted with increasing error value (e_i) and that the first level ($i = 1$) refers to accurate execution. Assuming that the set $\mathbf{L}$ contains all the wires of the accurate netlist, the optimization problem of producing our approximate circuits with dynamically reconfigurable accuracy can be formulated as follows:

$$\begin{aligned} &\underset{i\in[1,N]}{\text{minimize}} \left(Power(\mathbf{Y_N}|i)\right) &(3)\\ &\text{subject to } Error(\mathbf{Y_N}|i) \le e_i, \quad \forall i \in [1, N]\\ &\mathbf{Y_N} = \bigcup_{i=1}^{N} \mathbf{X_i},\\ &\mathbf{X_i} = \{(w_{i_1}, s_{i_1}), \ldots, (w_{i_z}, s_{i_z})\},\\ &\forall j \quad (w_j, s_j) \in \mathbf{L} \times \{ASW0, ASW1\}, &(4)\end{aligned}$$

where $Power(\mathbf{Y_N}|i)$ and $Error(\mathbf{Y_N}|i)$ are respectively the power consumption and the error value of the approximate circuit when operating at the accuracy level i. The tuples $(w_j, ASW0)$ and $(w_j, ASW1)$ are mutually exclusive, i.e., only one of them can belong in $\mathbf{Y_N}$. However, a tuple (w_j, s_j) can belong to more than one sets $\mathbf{X_i}$. To asses the complexity of our optimization problem, we rewrite (3) as:

$$\underset{i\in[1,N]}{\text{maximize}} \left(\frac{1}{Power(\mathbf{Y_N}|i)}\right). \qquad (5)$$

This is a combinatorial optimization problem and is mapped to the multiple nonlinear Knapsack problem [50]. Thus, it is NP-hard as a generalization of the single linear knapsack problem [50]. In our case, the cost (error) and value (power reduction) functions take real values, are non linear, and can be nonconvex or nonconcave. Moreover, the cost and value of selecting an item (replace a wire with ASW0 or ASW1) depend on the already selected items. To further elucidate the complexity of this problem, we note that the size of the design space is $3^{|\mathbf{L}|}$ (every wire can be either accurate or approximated by ASW0 or ASW1) and the evaluation of every design point requires very time consuming circuit simulations. Therefore, considering that even a simple adder circuit can feature has 292 wires (see Section IV), the design space explodes for more complex circuits. Hence, the increased complexity of this problem mandates the use of heuristic algorithms in practice.

Algorithm 1 Our RETSINA Optimization Algorithm

Input: 1. L: list of circuit wires, 2. E: list of the error bounds
Output: X: list of sets of tuples. The element X[i] refers to the set $\mathbf{X_i}$ of (7)

```
    # calculate the error value of the single approximation tuples
 1: P = []
 2: for all w in L:
 3:   for all s in {ASW0, ASW1}:
 4:     e=calcError([{(w,s)}])
 5:     P.append((w,s,e))
    # call the recursive function
 6: N = len(E)
 7: X = recSolver (P,E,N)
 8: return X
    # recursive function
 9: def recSolver (P,E,i):
10:   if N == 1:
11:     return []
12:   else:
13:     X = recSolver (P,E,i−1)
14:     P_i = []
15:     for all (w,s,e) in P:
16:       if e ≤ α× E[i]:
17:         P_i.append((w,s))
18:     X = SimulatedAnnealing (P_i, E[i], X, i)
    # SimulatedAnnealing is described in Algorithm 2
    # SimulatedAnnealing calculates X[i] and may modify X[j] ∀ j<i
19:     return X
```

A. PROPOSED HEURISTIC OPTIMIZATION

To solve our optimization problem we first reformulate it and then we employ the heuristic optimization presented in Algorithms 1 and 2. Simulated Annealing [51] is used in our heuristic approach to extract a good enough solution in a fixed and reasonable amount of time. The optimization decision is modified and reformulated as follows:

$$\begin{aligned} &\text{minimize } \left(\sum_{i=1}^{N} \left(p_i \times Power(\mathbf{Y_N}|i)\right)\right) &(6)\\ &\text{subject to } Error(\mathbf{Y_N}|i) \le e_i, \quad \forall i \in [1, N]\\ &\mathbf{Y_N} = \bigcup_{i=1}^{N} \mathbf{X_i},\\ &\mathbf{X_{i-1}} \subseteq \mathbf{X_i}, \quad \forall i \in [2, N]\\ &\mathbf{X_i} = \{(w_{i_1}, s_{i_1}), \ldots, (w_{i_z}, s_{i_z})\},\\ &\mathbf{X_1} = \{\},\\ &\forall j \quad (w_j, s_j) \in \mathbf{L} \times \{ASW0, ASW1\}, &(7)\end{aligned}$$

where p_i denotes the probability that the accuracy level i is selected at runtime. Without loss of power efficiency the optimization goal (3) is replaced by (6). In other words, instead of minimizing the power consumption of each accuracy

Algorithm 2 Simulated Annealing Pseudocode

Input: 1. P_i: list of potential tuples, 2. errorBound: error constraint 3. X: current solution (for i-1), 4. i: accuracy level
Output: R: list of sets of tuples (X is of size i-1 and R is of size i)

```
 1: Initialize temperature T
    # initial state X[i] = X[i−1]
 2: Let R = X # R is the current solution state
 3: R[i] = X[i−1]
    # first phase: build initial solution by adding random elements
 4: for b=0 to $\beta\times$netlistSize:; # netlistSize=|L|
 5:   if length(R[i]) == numSwitches: break
 6:   newSolutionSet = []
 7:   for all d < $\delta$:;# parallel execution
 8:     Pick random (w,s) in $P_i$-R[i], $R_{new}$ ← add (w,s) in R[i]
 9:     evaluate error and energy of $R_{new}$
10:     prob ← eval prob(Energy(s),Energy($s_{new}$),T)
11:     if prob ≥ rnd(0, 1) & checkError & checkDelay:
12:       newSolutionSet.append($R_{new}$)
13:   R ← select $R_{new}$ with the minimum energy in newSolutionSet
14:   if epoch length reached & Error(R) increased: update T
    # second phase: neighbor evaluation
15: for b=$\beta\times$netlistSize to $(\beta+\gamma)\times$netlistSize:
16:   newSolutionSet = []
17:   for all d < $\delta$:;# parallel execution
18:     Pick random (w,s) in R[i]
19:     select random (w′,s′) s.t. w′ in neighbors(w) and (w′,s′) in $P_i$
20:     $R_{new}$ ← remove (w,s) from R[j] ∀ j≤i, add (w′,s′) in R[i]
21:     prob ← eval prob(Energy(s),Energy($s_{new}$),T)
22:     if prob ≥ rnd(0, 1) & checkError & checkDelay:
23:       newSolutionSet.append($R_{new}$)
24:   R ← select $R_{new}$ with the minimum energy in newSolutionSet
25:   if epoch length reached & Error(R) increased: update T
26:   if Energy(R) is constant for $\tau$ epochs: return R
27: return R
```

level i in isolation, we aim in minimizing the average power consumption with respect to the overall operation of the circuit. If the values p_i are not known, we assume that all the accuracy levels are equiprobable. Furthermore, we assume that $\mathbf{X_{i-1}} \subseteq \mathbf{X_i}$, $\forall i \in [2, N]$, i.e., the approximations selected for the accuracy level i-1 are also selected in the accuracy level i. Although this decision might lead to some efficiency loss, it significantly reduces the complexity of our optimization decision. In order to compute the set $\mathbf{X_i}$, the solution obtained for the set $\mathbf{X_{i-1}}$ is reused and we do not need to initiate the solution search from an empty set. Note that since $e_{i-1} < e_i$, the set $\mathbf{X_{i-1}}$ satisfies, by definition, the error constraint of the level i. Moreover, leveraging that $\mathbf{X_{i-1}} \subseteq \mathbf{X_i}$, we obtain a very simple implementation for the circuitry that dynamically controls the accuracy levels, i.e., if the accuracy level i is selected, then all the switches of the sets $\mathbf{X_j}$ with $j \leq i$ are activated. For N accuracy levels, a N-1 bit input control signal is required to select the accuracy level at runtime. Every bit j of the input control signal is used to activate the switches that belong in the set $\mathbf{X_j} - \mathbf{X_{j-1}}$. Thus, to select the accuracy level i at runtime, the input control signal has to be set to 2^{i-1}-1. Next, to further reduce the complexity of our problem and without any loss of optimality, we prune the design space of the possible tuples (w_j, s_j) that can be selected in each set $\mathbf{X_i}$. To achieve this, the error value of all the tuples $(w, s) \in \mathbf{L} \times \{ASW0, ASW1\}$ (when are applied individually) is computed. Then, for each set $\mathbf{X_i}$, we consider only the tuples that result to an error value less or equal to $\alpha \times e_i$. In this work, we set $\alpha = 2$, but lower values can be used to further prune the design space.

The size of the final approximation set, i.e., $|\mathbf{Y_N}|$, mainly defines the overheads induced by the proposed technique. Without loss of generality, assume that ASW0, ASW1 feature the same area (A_{ASW}), capacitance (C_{ASW}), and delay (D_{ASW}). The area overhead is proportional to the number of switches [52] and can be bound by $|\mathbf{Y_N}| \times A_{ASW}$, i.e., $O(|\mathbf{Y_N}|)$. Similarly, considering (2) and that the area increase is bounded, the power overhead when operating at the accurate level is also $O(|\mathbf{Y_N}|)$. However, these overheads might be constrained by the user's or the system's requirements. Therefore, to control the power and area overheads at design time, we use the configuration parameter *numSwitches* (Algoritm 2) that limits the maximum number of switches that will be used, i.e., $|\mathbf{Y_N}| \leq numSwitches$. Moreover, if a switch replaces a wire in a path, the path's delay will increase when operating at the accurate level. Hence, the increase of the path's delay can be estimated by $k \times D_{ASW}$ [52], where k is the number of switches in that path. The gate-level netlist of a combinational circuit can be viewed as a direct acyclic graph (DAG). The circuit gates are the graph nodes and the wires are the edges. By parsing the circuit's DAG we can measure the number of added switches per path. Using PrimeTime we perform a STA and extract the delay of all the paths from the circuit's inputs to its outputs. Then, the DAG is annotated with the delays computed by STA and the value D_{ASW} is obtained from the technology library. As a result, using the annotated DAG we can calculate and constraint the delay increase by limiting accordingly the number of switches per path. For example, in case that we do not want to increase the circuit's critical path delay (CPD), assuming that PD is a path's delay obtained from STA, up to $\lfloor (CPD - PD)/D_{ASW}) \rfloor$ switches are allowed in this path.

The recursive Algorithm 1 is used to extract a solution of the modified problem (6) subject to (7). Considering that $\mathbf{Y_N}$ is the final solution (N accuracy levels), Algorithm 1 computes first the solution for the problem with N-1 accuracy levels, i.e., $\mathbf{Y_{N-1}} = \bigcup_{i=1}^{N-1} \mathbf{X_i}$. Then, $\mathbf{Y_{N-1}}$ is used to calculate $\mathbf{X_N}$ and thus extract the final solution $\bigcup_{i=1}^{N} \mathbf{X_i}$. In each recursion i, Algorithm 1 is allowed to i) remove tuples from the sets X_j with $j < i$ and ii) add new tuples (i.e., tuples that are not already selected) to X_i. Given the sets $\mathbf{X_1}, \ldots, \mathbf{X_{i-1}}$, Simulated Annealing [51] (SA) is used to calculate $\mathbf{X_i}$. Algorithm 2 presents the pseudocode of our Simulated Annealing implementation. SA is divided in two phases. In the first phase, random tuples are added in the solution until reaching the *numSwitches* threshold. To achieve this, Algorithm 2 performs a random sampling over the entire design space of the potential tuples and creates the initial solution state. SA selects a random tuple $(w, s) \notin \mathbf{X_i}$ and checks if the new state $\mathbf{X_i} \bigcup \{(w, s)\}$ will be accepted. The new state is accepted only if i) the acceptance probability function of SA, ii) the error constraints, and iii) the delay constraint are satisfied. This phase terminates when a maximum iteration count (line 4) or when the the maximum number of switches *numSwitches* (line 5) are reached. In the second phase, SA evaluates the neighbor states of the current state. Algorithm 2 selects a random tuple $(w, s) \in \mathbf{X_i}$ and then picks a random neighbor w' of w. Then, it replaces (w, s) with $(w', s') = (w', random(ASW0, ASW1))$ to generate the new state under consideration. Similar to the first phase, the new state is evaluated to be accepted or not. If $(w, s) \in \mathbf{X_j}, \forall j < i$, it is removed from $\mathbf{X_j}$ and (w', s') is added only to $\mathbf{X_i}$, i.e., $\mathbf{X'_j} = \mathbf{X_j} - \{(w, s)\}, \forall j < i$ and $\mathbf{X'_i} = \mathbf{X_i} \bigcup \{(w', s')\} - \{(w, s)\}$. The tuple (w, s) is removed from $\mathbf{X_j}$ so that the constraint $\mathbf{X_{i-1}} \subseteq \mathbf{X_i}$ is satisfied. Hence, the new state under consideration is given by $Y'_i = \bigcup_{j=1}^{i} \mathbf{X'_j}$. When referring to the neighbors of a wire w, we refer to all the wires that are either inputs or outputs of the gates that the wire w connects. The neighbors of a wire w are therefore obtained by the circuit's DAG. In Algorithm 2, (6) is used as the energy function of SA, i.e., $E = \sum_{j=1}^{i} \left(p_j \times Power(\mathbf{Y_i}[j])\right)$. Note that, in each epoch, SA tries to reduce its energy function. As a cooling schedule we use an adaptive variant of the exponential multiplicative cooling [51]. At the end of each epoch, if the error value decreased, the temperature remains constant. Otherwise, the temperature is updated (multiplied by 0.9). In order to limit the execution time of SA, its maximum iteration count is set to be proportional to the number of wires of the circuit's netlist, i.e., $(\beta+\gamma)\times|\mathbf{L}|$. Specifically, in this work we use $\beta = 10$ and $\gamma = 100$. Moreover, if the energy remains constant after τ epochs, we consider that SA converged, Algorithm 2 terminates, and the current state is returned. In our work, we set $\tau = 500$ to allow SA considerable time to escape a local minimum. In each iteration of SA, δ new states are evaluated in parallel. We set $\delta = 80$, i.e., the number of available hardware threads in our system. The parameter δ determines also the epoch length of SA. Considering the largest size of the examined circuits and that $\delta = 80$ we set the epoch length to be 25. The execution time of each epoch is determined by the time required for evaluating the energy and error values of a state. To further optimize the SA execution time, memoization is used, i.e., the error and energy values of every evaluated state are stored in a dictionary. Each time the same state occurs again, the stored values are reused instead of recomputing them. This is particularly useful to speedup the execution time and especially to avoid recomputing the energy and error values when removing a tuple from a set $\left(\text{e.g., } \mathbf{X'_j} = \mathbf{X_j} - \{(w, s)\}\right)$.

However, computing the energy and error values in Algorithm 2 requires: i) netlist modification (replace each wire w_j with the switch s_j), ii) circuit simulation (calculate the error value and generate the circuit's switching activity), and

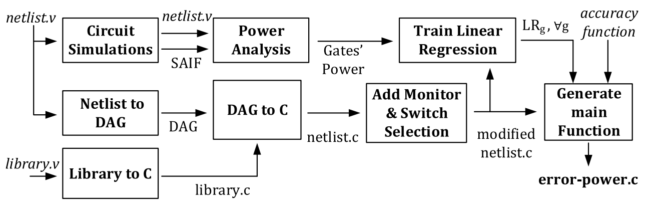


FIGURE 7. The flow diagram of the implemented gate-level to C converter. The resulting C function computes the circuit's error value and estimates its power consumption for a given approximation set and an input dataset.

iii) power analysis (calculate the power consumption using the obtained switching activity). Therefore, despite the aforementioned optimizations, the execution time of the proposed heuristic solution is defined by the increased time required for a circuit simulation. To avoid performing circuit simulations, we implement a gate-level netlist to C converter and evaluate the circuit's error and power values at C-level.

Note that Simulated Annealing (SA) is selected in Algorithm 2 because our optimization problem could be directly mapped to it and because SA enables the extraction of a good enough solution in a fixed amount of time [51]. However, our framework is not strictly bound to SA and our gate-level to C converter (Section III-B) enables the exploitation of any other heuristic algorithm (with similar objective) to replace SA. All the parameters of SA, e.g., epoch length and cooling schedule, as well as the parameters $\alpha, \beta, \gamma, \delta$ and τ can be set accordingly by the user. In our evaluation, these parameters are empirically set to the aforementioned values to provide SA reasonable time to converge, but also limit the execution time of our framework to reasonable values.

B. PROPOSED C-LEVEL POWER AND ERROR EVALUATION

Fig. 7 presents the flow employed to enable very fast power and error evaluation of the produced approximate circuits at C-level. Our gate-level to C converter is used to translate the circuit's gate-level description to a C function and thus, enable fast circuit simulations at C. During the C-level simulations the circuit's outputs are computed and its power consumption is estimated. An analytical power model is used to estimate the power consumption with respect to the togling activity of the circuit's wires. Hence, using i) the provided accuracy function, ii) the gate-level to C converter, and iii) the power model we obtain the circuit's error and power values.

Power Model: The power consumption of a circuit can be approximated by the sum of the power consumption of the circuit's gates [49], [53]:

$$Power(circuit) = \sum_{\forall gate \in circuit} \left(Power(gate)\right). \tag{8}$$

Similar to (1)-(2), the power consumption of a gate is estimated by [48]:

$$Power(gate) = P_{static} + aCV^2 f. \tag{9}$$

Therefore, a linear regression model can be used and efficiently estimate a gate's power consumption with respect to its switching activity.[2] The intercept of the linear regression comprises the static power consumption of the gate and the product slope×switching activity represents the gate's dynamic power consumption. For each gate in the circuit, we train a linear regression and estimate the gate's power with respect to the toggling ratio (i.e., toggling count over input dataset size) of its outputs. Assuming that LR_g and t_g are respectively the linear regression model and the toggling ratio of the outputs of a gate g, the circuit's power consumption is estimated by:

$$Power(circuit) = \sum_{\forall g \,\in\, circuit} \big(LR_g(t_g)\big). \qquad (10)$$

As a result, after training the linear regression models LR_g, we only need to calculate t_g to obtain the circuit's power consumption. Using the gate-level netlist to C converter, t_g is estimated at C-level by counting the number of times that each variable's value is changed. Thus, the circuit's power consumption can be estimated at C-level and we do not need to perform vast circuit simulations and power analyses during the proposed heuristic optimization. To compute the LR_g models we simulate the circuit with different (small) input datasets and generate varying switching activity files (SAIFs). Next, using Synopsys PrimeTime, power analyses are performed for every SAIF and the power consumption of each gate is extracted. In addition, for the same input datasets, C-level simulation is performed and we obtain the respective t_g values. Using the computed toggling ratios (t_g) and the power values, a linear regression (LR_g) is trained to estimate the power of each gate. Circuit- and C-level simulations might result to different t_g values and for this reason we train the LR_g with respect to the t_g obtain by the C simulation. All the gates of our examined technology library (Section IV) feature a single output wire and thus, only two circuit simulations are performed to train the LR_g. However, even for libraries that contain gates with several outputs, the number of the required simulations is very limited. For the most complex circuit we examine in Section IV (i.e., 3769 gates), training the linear regression models required less than 10 minutes (with 40,000 inputs per simulation). Similarly, we also train to two Linear Regression models LR_{ASW0} and LR_{ASW1} to estimate the power consumption of the approximate switches ASW0 and ASW1. Then, when a switch is used to replace a wire, the respective model (LR_{ASW0} or LR_{ASW1}) is added to the sum of (10). When the switch is off, its toggling ratio equals the respective one of the replaced wire. When it is on, its toggling ratio is zero.

Gate-Level to C: The gate-level netlist to C converter is implemented to translate the circuit's Verilog netlist to a C function and accelerate thus our optimization Algorithm 2. The converter parses the Verilog file of the technology library and for each gate in the library, it generates an equivalent C function (with the same name and arguments). The Verilog gate primitives are replaced with the respective bitwise operators in C, e.g., "and" is replaced by the C operator '&'. Pointers are used for the function outputs so that every function can seamlessly have more than one output values. Translating the library from Verilog to C needs to be performed only once per technology library. Note that in this work we target combinational circuits and thus, the library's sequential cells are not translated. Then, given the C description of the library, the gate-level netlist of the circuit is transformed to its C equivalent. The parser reads the Verilog description of the netlist and generates the circuit's direct acyclic graph (DAG). Then, the DAG is traversed and a C function equivalent to the gate-level-netlist is generated. The DAG is parsed level by level to build the C function since C statements are executed in order (unlike the gate-level Verilog statements that are concurrent). During this procedure, the DAG nodes (gates) are replaced by the respective functions calls to the library and the edges (wires) are used as the function's input and output variables. Next, the generated C function is modified as follows: i) each variable (i.e., circuit's wire) is monitored to count its toggling and ii) a case statement is used for each variable to select between "ASW0", "ASW1" or the variable's value (i.e., not approximated). After generating the modified C function, the main function is produced. The inputs of the main function are i) a dataset file that contains the inputs for the circuit simulation and the respective accurate outputs and ii) an array containing the indexes of the approximated wires and the respective switch type. The output of the main function is the circuit's error and its power consumption estimation. The main function reads the inputs from the file and calls the circuit's modified C function to compute the approximate outputs. Then, it compares the obtained outputs with the accurate ones (read from the file) and computes the circuit's error value with respect to the provided accuracy function. Moreover, using the toggling count measured by the monitor function and the power model (10), it produces the power estimation. Therefore, given an approximate set $\mathbf{Y_i}$, using the gate-level netlist to C converter we can obtain, in a very fast manner, the values $Error(\mathbf{Y_i}|i)$ and $Power(\mathbf{Y_i}|i)$ required in our optimization process. Algorithms 1 and 2 are written in Python and ctypes is used to run C-level circuit simulations.

Efficiency Evaluation: In Fig. 8 we evaluate the efficiency, in terms of power estimation and execution time, of performing C-level simulations instead of gate-level ones using QuestaSim. First, we examine the efficiency of the adopted power model (10). Simulated annealing aims to minimize its energy function. To achieve this, it mainly moves from the current state to a neighbor state that features less energy. As a result, in our optimization (Algorithms 1 and 2) precisely estimating the order of the approximate circuits with respect to their power consumption is of high significance. Hence, to assess the efficiency of our power estimator we use the Spearman's rank correlation coefficient, i.e., we examine how efficiently the power ranking of our approximate circuits is estimated. The Spearman's rank correlation coefficient takes

[2]The static power is given by $P_{static} = I_{leakage} \times V$ [48] and thus, it does not depend on the switching activity.

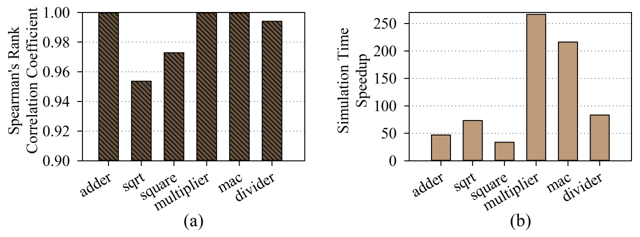


FIGURE 8. Evaluation of the efficiency of our gate-level to C converter in terms of (a) estimating the power ranking of our approximate reconfigurable circuits and (b) accelerating the simulation time required to compute a circuit's outputs. (a) and (b) are reported for the circuits examined in Section IV. As the baseline, we consider the simulation time of the circuit simulation using QuestaSim and the power consumption obtained by PrimeTime.

values in the segment [-1,1], where the value 1 refers to perfect positive correlation. For all the benchmark examined in Section IV, we generate 50 random approximate counterparts (300 designs in total), i.e., we randomly replace some circuit wires with the switches ASW0 or ASW1. To measure their power consumption (at the approximate operation) we perform circuit-level simulations using Mentor QuestaSim and power analyses using Synopsys PrimeTime. Then, we estimate their power consumption by performing C-level simulations using our gate-level to C converter and the proposed power model (10). For each circuit, the obtained Spearman's rank correlation coefficient is illustrated in Fig. 8a. On average, the Spearman's rank correlation coefficient is 0.987, i.e., almost perfect positive correlation, and ranges from 0.95 up to 1. Hence, the proposed power model efficiently estimates the power ranking of our approximate circuits. In addition, we also examine the accuracy of our power model by examining the Root Mean Square Error (RMSE) between the values predicted by our power model and the values calculated by PrimeTime. For all the examined circuits in Fig. 8a (300 in total), the RMSE value is 9.22×10^{-5}. Hence, considering that the power consumption is in the order of mW, our C-level power estimation delivers also accurate enough power estimation. Next, we evaluate the efficiency of the gate-level to C converter in accelerating the execution of our optimization process. For the benchmarks examined in Section IV, we measure the speedup achieved by performing circuit simulations at C-level using the proposed gate-level to C converter compared to performing simulations using QuestaSim. For every circuit, 10^5 random generated inputs are used and every experiment is conducted 3 times to measure the average execution time. The speedup achieved by performing C-level simulations is reported in Fig. 8b. The average attained speedup is 120× and ranges from 34× up to 266×. Considering the vast number of error and power evaluations performed in Algorithms 1 and 2, the gate-level to C converter significantly accelerates their execution.

IV. EXPERIMENTAL EVALUATION

In this section, we evaluate the efficiency of the proposed framework using industry-strength tools and benchmarks. We use two state-of-the-art technology standard cell libraries with differing characteristics: a conventional 7nm FinFET library [54] and its counterpart using Negative Capacitance Field-Effect Transistors (NCFET) [55], [56]. Six benchmarks are considered in our evaluation, i.e., adder, square root (sqrt), square, multiplier, multiply-accumulate (mac), and divider. All the benchmarks are obtained from the industry-level Synopsys DesignWare library (M-2016.12) and they are listed in Table 1. All the circuits are synthesized targeting performance optimization and they are compiled using the "compile_ultra" command of Design Compiler. Moreover, all the evaluated circuits are synthesized and simulated at their critical path delay, i.e., with zero slack. As an accuracy function, the Mean Relative Error Distance (MRED) [14] is used. MRED is the average relative error and is calculated by $\frac{1}{K}\sum_{1}^{N}|\frac{O_A-O_E}{O_E}|$, where O_A, O_E are the approximate and accurate outputs, respectively, and K is the total number of outputs. For each benchmark, three accuracy levels are considered with MRED bounds {0%, 1%, 2%}. The accuracy level 1 refers to accurate execution (noted as MRED bound 0%) while accuracy levels 2 and 3 target 1% and 2% MRED, respectively. Circuit simulations using QuestaSim are performed to calculate the output/error of the generated circuits. PrimeTime is used for power analyses and to compute power consumption of all the circuits. After generating the approximate circuits, their area and delay values are calculated using Design Compiler and PrimeTime, respectively. For each circuit, two distinct randomly generated input datasets are used. The first one (2×10^5 inputs) is used in the optimization phase of RETSINA. The second one (10^6 inputs) is used for the circuit simulations. For each circuit, the parameter *numSwitches* is set to be equal to 5% of the number of the circuit's gates. In our evaluation, we examine a worst case analysis scenario, i.e., fully optimized designs synthesized at their critical path delay. Therefore, since the circuit paths are well balanced, we allow RETSINA to add switches to the circuit's critical paths in order to increase the approximation candidates. However, we constraint the delay overhead to be at most equal to the delay of an approximate switch. In this worst case evaluation, all the accuracy levels are considered equiprobable, i.e., operating at the accurate level is highly probable. Finally, since the circuits are operated at their critical path delay, if in Algorithm 2 the delay increases, we update the slopes of the LR_g models to reflect the new frequency. The frequency is given by $1/delay$. Thus, to update the slopes we have to multiply them by d/d', where d is the delay of the accurate circuit and d' is the delay of the approximate one. The experiments are run on a dual Xeon Gold 6138 server that features 80 threads and 128GB RAM.

Error Analysis: First, the error characteristics of the approximate circuits generated by RETSINA are examined. Each accurate circuit of Table 1 is synthesized targeting the 7nm FinFET library to obtain the accurate netlist. Then, RETSINA is used to generate its dynamically reconfigurable approximate counterpart. Note that identical results are obtained for the NCFET library since the error optimization in our framework is technology independent and the

TABLE 1. The benchmarks used to evaluate the proposed framework. All the benchmarks are from Synopsys DesignWare library.

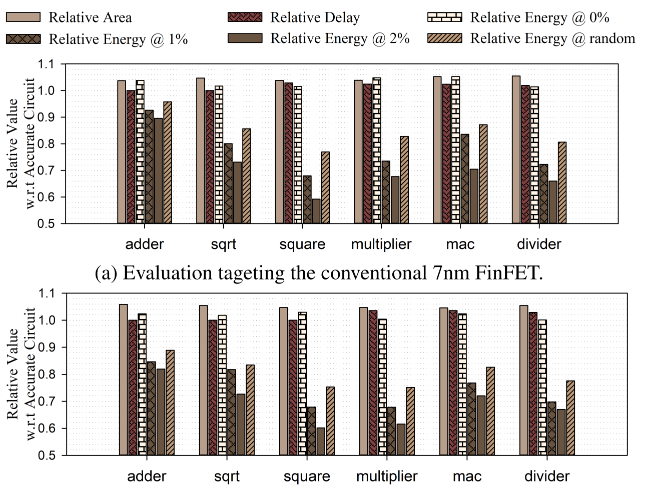
Circuit	Operation	Area (um^2)	Num. of Gates	Num. of Wires
adder	17b $\leftarrow$ 16b+16b	32	256	292
sqrt	8b $\leftarrow$ $\sqrt{16b}$	40	431	456
square	32b $\leftarrow$ (16b)^2	92	904	925
multiplier	32b $\leftarrow$ 16b $\times$ 16b	148	1418	1465
mac	33b $\leftarrow$16b$\times$ 16b + 32b	175	1578	1646
divider	16b $\leftarrow$ 16b/8b	323	3769	3796

TABLE 2. Error evaluation of the approximate circuits generated by RETSINA.

Circuit	accuracy level 1		accuracy level 2		accuracy level 3	
	MRED	MAD	MRED	MAD	MRED	MAD
adder	0.00%	0.00%	0.72%	0.40%	1.43%	0.90%
sqrt	0.00%	0.00%	0.94%	0.50%	1.84%	1.20%
square	0.00%	0.00%	0.95%	0.10%	1.98%	0.30%
multiplier	0.00%	0.00%	1.12%	0.20%	1.91%	0.40%
mac	0.00%	0.00%	0.69%	0.00%	1.86%	0.20%
divider	0.00%	0.00%	0.99%	0.20%	1.91%	0.60%

induced error is controlled at design time. To compute the error value at each accuracy level, 3 circuit simulations are performed for every generated approximate circuit. In each simulation the approximate circuit is operated at a single accuracy level. In Table 2, the error characteristics of the generated approximate circuits are reported. As shown in Table 2, the target MRED constraint is well satisfied in all the cases. Moreover, as a measure of dispersion, the Median Absolute Deviation (MAD) of the Relative Error Distance (RED) is also presented in Table 2. MAD is a robust measure of statistical dispersion and is defined as the median of the absolute deviations from the distribution median. MAD retains units and is reported as a percentage since RED is also reported as percentage. As presented in Table 2, MAD takes very small values (with respect also to MRED). For the accuracy level 2, i.e., 1% MRED bound, the average MAD is 0.23% ranging from 0.0% up to 0.50%. Similarly, for the target accuracy level 2 (2% MRED bound) MAD is on average 0.60% and ranges from 0.20% up to 1.20%. Hence, the RED value of the generated approximate circuits features very small dispersion and is well concentrated close to the target error value.

Hardware Analysis targeting the 7nm FinFET Conventional Technology: In Fig. 9a the hardware efficiency of the generated reconfigurable approximate circuits is evaluated when targeting the 7nm FinFET library [54]. Fig. 9a presents their area, delay, and energy characteristics. The energy consumption is reported for each accuracy level as well as in a random case where the accuracy level is randomly selected and all the accuracy levels feature the same probability to be selected. The examined metrics are reported as relative values with respect to the ones of the corresponding accurate design. The energy consumption is calculated by the product power$\times$delay, where the power and delay values for each circuit are computed using PrimeTime. In Fig. 9a, RETSINA replaced 11, 18, 36, 62, 78, and 178 wires with approximate switches in the adder, sqrt, square, multiplier, mac, and divider circuits, respectively. Therefore, the *numSwitches* constraint we set, is well satisfied. On average, the area



(a) Evaluation tageting the conventional 7nm FinFET.

(b) Evaluation tageting the emerging technology 7nm NCFET.

FIGURE 9. Evaluation of the area, delay and energy of the proposed reconfigurable approximate circuits. The examined metrics are presented as a relative values with respect to the corresponding accurate design. The relative energy is reported for each accuracy level as well as for random accuracy level selection. The notation @0%, @1%, and @2% refers to the accuracy levels 1, 2, and 3 with MRED bounds 0%, 1%, and 2%, respectively. The notation @random refers to random selection of the accuracy level at runtime, with all the accuracy levels being equiprobable. All the approximate and accurate circuits are implemented targeting (a) the conventional 7nm FinFET technology and (b) the NCFET one.

overhead imposed by the added switches is 4.4% and ranges from 3.7% up to 5.5%. Hence, the induced area overhead is well constrained by the parameter *numSwitches*. Note that, in our evaluation *numSwitches* is set to 5% of the number of the circuit's gates. To explicitly force the area overhead below *Z*% the user should select *numSwitches*= $\lfloor Z\times$(circuit area/ max area(ASW0,ASW1)/100)$\rfloor$. Similarly, the delay overhead is 2% on average and ranges from 0% up to 2.9%. For the small circuits (adder, sqrt) there is no delay overhead, i.e., they operate at same frequency as the accurate design. For the larger benchmarks (square,multiplier, mac, divider) the delay overhead is 10ps, i.e., RETSINA's delay constraint is satisfied. As the circuit complexity increases, more wires are approximated. The examined designs are fully optimized and their paths are well balanced. Therefore, as the number of approximated wires increased, RETSINA couldn't avoid to approximate any of the critical paths wires. However, this delay overhead is very small and the tradeoff between delay increase and energy efficiency can be controlled at design time. Furthermore, for each accuracy level, the energy consumption of the generated circuits is evaluated. As expected, when the approximate circuits operate at the accuracy level 1 (i.e., accurate execution), the energy consumption slightly increases due to the induced circuitry. On average it is 3% and ranges from 1.4% up to 5%. The aforementioned overheads originate from the addition of the approximate switches and thus, they constitute the cost of the control circuitry that is required to enable the dynamic accuracy reconfiguration at runtime. On the other hand, when the circuit operates at an approximate mode, significant energy savings are delivered. At the accuracy level 2 (1% MRED bound), the attained energy reduction is on average 22% and ranges from 7% up to 32%. Similarly, at the accuracy level 3 (2% MRED bound), the average energy reduction is 29%, ranging from 11% up to 41%. As shown in Fig. 9a, for higher MRED bounds the energy reduction is higher since more wires can be approximated and thus, RETSINA can further limit the circuit's switching activity. In addition, as the circuit complexity increases, higher energy reduction is also achieved. For example, for 2% MRED bound, the energy reduction increases from 11% for the adder circuit to 41% for the square. For complex circuits, that feature a lot of wires, the impact on the output value of approximating a wire is less significant compared to smaller circuits. Therefore, more wires can be approximated and higher energy savings are delivered. However in Fig. 9a, this scaling does not continue for the multiplier, mac, and divider circuits that feature 32%, 30%, and 34% energy reduction respectively. These circuits are affected by the tight delay constraint set in our optimization and thus, they couldn't exploit the full spectrum of the applied approximations. Nevertheless, even though the approximations applied on these circuits were limited by the delay constraint, they achieve significant energy savings. As shown in Fig 9a, the proposed reconfigurable approximate circuits feature an insignificant delay overhead that can be controlled at design time. Moreover, although they induce a small energy overhead when operating at the accurate level, they deliver significant energy savings when operating at an approximate mode. Considering the high energy gains achieved at the accuracy levels 2 and 3, the energy overhead of the accuracy level 1 is assumed negligible. Note that, this overhead affects the circuit's energy consumption only when accurate computations are required. Since approximate computing targets error resilient applications, the time spent in computations that require accurate execution can be considered insignificant compared to the overall execution time. Therefore, the impact of the of this small energy overhead on the total energy consumption is minimum. However, as a worst case scenario, we consider all the accuracy levels to be equiprobable. As shown in Fig. 9a, when the accuracy level is randomly selected, the average energy saving is 15%, ranging from 4% up to 23%. Note that for this random level evaluation, the MRED value is on average 0.92%. Specifically, the MRED of the adder, sqrt, square, multiplier, mac, and divider is 0.72%, 0.93%, 0.97%, 0.96%, 0.99%, and 0.96%, respectively. This random accuracy level evaluation demonstrates the high efficiency of the proposed framework. Despite performing accurate computations in the 33% of the time, significant energy savings are still obtained (15% on average), while the MRED value is very small (less than 1%).

Hardware Analysis targeting the 7nm NCFET Emerging Technology: In Fig. 9a we applied the proposed framework on varying benchmarks and showed that it is circuit independent. In Fig. 9b, we evaluate the efficiency of the proposed framework using a different technology library, showing that it is also technology independent and the delivered energy gains are retained. In Fig. 9b the efficiency of the proposed reconfigurable approximate circuits is examined when targeting the NCFET emerging technology library. Moreover, through this analysis, we evaluate, for the first time, the impact of approximate computing on NCFET-based circuits.

Negative Capacitance Field-Effect Transistor (NCFET) is very rapidly emerging as a promising technology for future computing [57], [58]. NCFET enables transistors to switch much faster without the need to increase the operating voltage. This is achieved, for the first time, through incorporating a thin layer of a ferroelectric (FE) material inside the transistor gate stack. The FE layer manifests itself as a negative capacitance that magnifies the applied vertical electric field leading to an internal voltage amplification, instead of a voltage drop as is the case in any existing conventional CMOS technology. Such an internal voltage amplification significantly improves the electrostatic integrity of transistor, leading to larger ON current and thus a higher switching speed, while the operating voltage still remains the same [55]. In short, the presence of a negative capacitance inside the transistor's gate results in a much larger current that enables a higher switching speed and as a result the circuit can be clocked at a higher frequency while the voltage remains the same. However, NCFET technology comes with a major drawback in which the presence of a negative capacitance increases the total gate's capacitance. *Therefore, compared to conventional CMOS technology, NCFET-based circuits will have a larger dynamic power as they will exhibit a larger total capacitance at the same V_{DD}.*

This, in turn, strongly reduces the efficiency of the NCFET technology and in the following we investigate, for the first time, how applying approximate-computing principles helps in eliminating the side-effects of NCFET, when it comes to the increase in the dynamic power. To achieve that, we employ our proposed framework to create reconfigurable-approximate NCFET circuits. Because our implementation is *technology independent*, we can directly plug in NCFET-aware standard cell libraries inside RETSINA to automatically generate reconfigurable approximate circuits in the NCFET technology. In this work, we leverage NCFET-aware cell library developed in [55], [56], that was created based on the 7nm FinFET technology node using a physics-based NCFET compact model. This allows us to have fair and direct comparisons with the conventional FinFET-based circuits.

In Fig. 9b, RETSINA added 11, 24, 42, 65, 66, and 178 switches in the adder, sqrt, square, multiplier, mac, and divider, respectively. Similar to Fig. 9a, the area overhead in Fig. 9b is 5% on average and ranges from 4.6% up to 5.8%. The delay overhead is 2%, on average, and the energy overhead when operating at the accurate level is on average 2%, ranging from 0.1% up to 3%. When operating at the accuracy level 2 (MRED bound 1%), the energy reduction is 25%, on average, and ranges from 15% up to 32%. When operating at the accuracy level 3 (MRED bound 2%), the energy reduction is 31% on average, ranging from 18% up to 40%. Finally, when the accuracy level is randomly selected, the energy

TABLE 3. Evaluation of the impact of the proposed framework on mitigating the energy efficiency downside of the NCFET based circuits.

Circuit	Implementation	Delay[a]	Energy[a]	Energy× Delay[a]
adder	FinFET accurate	1.00	1.00	1.00
	NCFET accurate	0.67	2.02	1.35
	NCFET RETSINA[b]	0.67	1.80	1.20
sqrt	FinFET accurate	1.00	1.00	1.00
	NCFET accurate	0.64	1.68	1.08
	NCFET RETSINA	0.64	1.40	0.90
square	FinFET accurate	1.00	1.00	1.00
	NCFET accurate	0.63	1.75	1.10
	NCFET RETSINA	0.63	1.31	0.83
multiplier	FinFET accurate	1.00	1.00	1.00
	NCFET accurate	0.62	1.70	1.06
	NCFET RETSINA	0.64	1.28	0.82
mac	FinFET accurate	1.00	1.00	1.00
	NCFET accurate	0.62	1.71	1.07
	NCFET RETSINA	0.64	1.42	0.91
divider	FinFET accurate	1.00	1.00	1.00
	NCFET accurate	0.60	1.67	1.01
	NCFET RETSINA	0.62	1.29	0.80

[a]Relative value with respect to the accurate FinFET based circuit.
[b]The accuracy level of the proposed approximate reconfigurable circuits is randomly selected and all the levels are equiprobable.

consumption is reduced by 20% on average, ranging from 11% up to 25%. In this case, the obtained MRED value is 0.97%, on average, and ranges from 0.7% up to 1.0%. Hence, our framework significantly reduces the energy consumption of the NCFET circuits. The energy savings in Fig. 9b circuits are even higher compared to Fig. 9a. The NCFET circuits exhibit larger capacitance and thus, they are benefited more by constraining the switching activity.

Next, we evaluate the impact of approximate computing in compensating the energy-efficiency downside of the NCFET based circuits. As shown in Table 3, the accurate NCFET circuits feature significantly higher performance (1.59× on average) compared to the respective FinFET based ones. However, this high performance comes at the cost of very high energy consumption. Compared to the accurate FinFET circuits, the energy consumption of the accurate NCFET ones is 76% higher, on average. Therefore, to asses the efficiency of the NCFET based circuits, we examine the Energy×Delay Product (EDP) metric. Despite the increased performance of the accurate NCFET circuits, their energy consumption is so high that eliminates the achieved delay gain. On average, the EDP of the accurate NCFET circuits is 11% higher than that of the FinFET ones. On the other hand, compared to the accurate FinFET based circuits, the approximate NCFET ones produced by RETSINA feature on average 1.56× higher performance and 9% smaller EDP. Note that, in Table 3, we consider the worst case scenario of Fig. 9b. The approximate NCFET circuits feature, mainly, smaller EDP than the respective accurate FinFET circuit. Only the EDP of the small 16-bit adder didn't decrease bellow the FinFET baseline. As aforementioned, in the 16-bit adder only a few wires are approximated due to the MRED and *numSwitches* constraints set in RETSINA. As a result, the achieved energy reduction is limited (11%) and thus, the EDP remained higher than the one of the FinFET based adder. However, for the rest

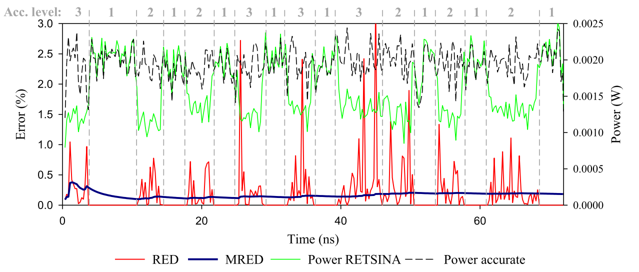


FIGURE 10. Runtime power comparison between the accurate square and its approximate counterpart produced by RETSINA. For the approximate square, the runtime variation of the RED and MRED is also presented. The accuracy level is randomly selected and 300 random generated inputs are considered.

circuits the EDP gain ranges from 9% up to 20%. As shown in Table 3, our framework enables the exploitation of NCFET based circuits by mitigating and in most cases eliminating their energy-efficiency drawback. Hence, very performant as well as energy efficient approximate circuits with dynamically reconfigurable accuracy are obtained.

Dynamic Reconfiguration: Fig. 10 presents an example of the dynamic accuracy reconfiguration. For this example, 300 random generated inputs are considered and the accuracy level is randomly selected. The square circuit is used as our driving circuit and similar results are obtained for the other circuits. The power variation, over time, of the approximate square is depicted in Fig. 10 and it is compared against the respective power variation of the accurate square. In addition, the RED and MRED variations are also illustrated in Fig. 10. The selected accuracy accuracy level is depicted above the graph. In this example, the average the energy reduction is 17.5% while the MRED value is only 0.20%. As shown in Fig. 10, selecting the accuracy level enables the user to dynamically control the RED value. Regarding the power variation, when the accuracy level 1 is selected, the approximate circuit features equal or a little higher power compared to the accurate one. However, when operating at levels 2 and 3, the approximate square features significantly less power than the accurate one. On average, as the level of approximation increases, larger distance between the two power graphs is observed and thus, higher power gain is achieved. The approximate circuits generated by RETSINA offer the user a very fine grain control of the applied approximation at runtime. Leveraging the on the fly accuracy control, the user can select the respective accuracy level, considering the input values and the application's state, and thus, effectively optimize the application's energy-error tradeoff.

Approximation Efficiency Evaluation: In Fig. 9 the energy efficiency of the circuits generated by RETSINA is evaluated. In Fig. 12, we examine the efficacy of the employed heuristic solution in identifying which wires will be approximated. RETSINA is compared against the state-of-the-art approximation technique GLP [21] that is illustrated in Fig. 11. Similarly to RETSINA, GLP is an automated approximate design framework that applies netlist approximations. However, GLP prunes the circuit's wires and thus, it does not support accuracy reconfiguration at runtime. Therefore, for

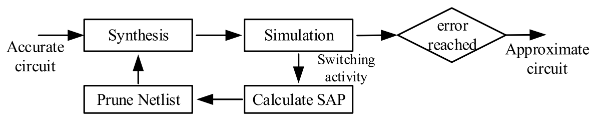


FIGURE 11. The flow diagram of the GLP framework [21].

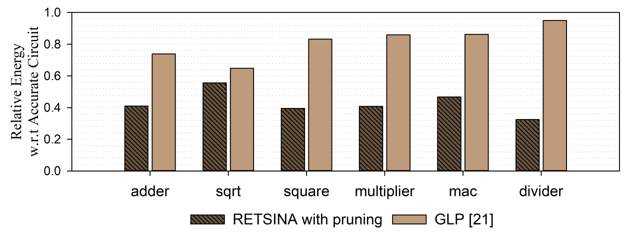


FIGURE 12. Evaluation of the approximation efficiency of the heuristic approach employed in the proposed RETSINA framework. The approximate circuits generated by RETSINA are compared against the ones produced by GLP [21]. All the circuits are generated targeting 1% MRED bound. For the fairness of the performed comparison, RETSINA also prunes the wires instead of replacing them with switches.

the fairness of the performed comparison, we examine only one accuracy level and in the last step of RETSINA instead of replacing the selected wires with the switches ASW0 and ASW1, we prune them with the respective value. Then, the obtained circuit is synthesized to remove any floating gates. All the approximate circuits in Fig. 12 are designed with MRED bound 1% and are synthesized and simulated at the critical path delay of the respective accurate circuit. Fig. 12 depicts the relative energy consumption of the approximate circuits generated by RETSINA and GLP with respect to the corresponding accurate one. On average, the approximate circuits generated by RETSINA feature 46% less energy consumption compared to the GLP circuits. This high energy gain demonstrates the approximation efficiency of our power-aware heuristic approach. GLP [21], in order to avoid the complexity of the circuit pruning optimization problem, uses an iterative greedy algorithm. At each iteration, GLP prunes the wire that features the lowest significance×activity product (SAP). GLP terminates when the error bound is reached. In each iteration, after pruning, the netlist is synthesized to generate the respective approximate one. Nevertheless, such a greedy approach does not ensure energy efficiency. During the pruning procedure, [21] considers only SAP and does not examine the impact of pruning a wire on the circuit's energy consumption.

In addition, GLP does not consider the error propagation in the performed pruning. For example, the error generated by pruning a wire might be compensated by pruning another one. However, by examining only the output significance of each wire, such information is not leveraged and thus, a fewer number of wires is pruned. On the other hand, RETSINA evaluates the circuit's error before selecting the wire to be approximated and thus, it is able to capture such cases. Furthermore, RETSINA uses the proposed power model and estimates the circuit's power consumption. Then, RETSINA selects the respective approximations that attains the highest power reduction. Hence, the proposed framework is able to deliver

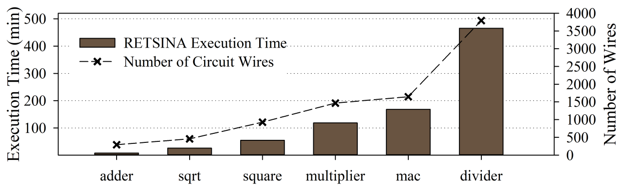


FIGURE 13. The execution time of the proposed framework with respect to the examined benchmarks that feature increasing circuit complexity. Three accuracy levels are considered for each circuit.

more energy-efficient approximate solutions compared to the greedy and power agnostic approach of [21].

Time Complexity Evaluation: Next, the time complexity of the proposed framework is evaluated. For every circuit examined, the time required by RETSINA to generate its reconfigurable approximate counterpart is reported in Fig. 13. The execution time of RETSINA ranges from 8 minutes for the adder that features 292 wires up to 7.76 hours for the divider, i.e., the largest examined circuit with 3796 wires. As aforementioned, the bottleneck of RETSINA's execution is the time required to evaluate the circuit's error and power. Hence, considering the speedup delivered by performing the proposed C-level error and power evaluation (Fig. 8), the generation of our reconfigurable approximate circuits would be infeasible without the proposed approach. As shown in Fig. 13, the execution time of RETSINA increases as the circuit complexity increases. This is expected since in Algorithm 2, we set the iteration constraint to be proportional to the number of the circuit's wires. The execution time required by RETSINA is comparable (or even smaller) to existing approximate design automation frameworks. For example, [32] requires 1.16 minutes for only a 4-bit adder, [21] reports 15-20 minutes for a 32-bit adder, [23] requires 4 hours for a 16-bit multiplier, and [26] needs 20 hours for an 8-bit multiplier. Note that these frameworks apply only fixed approximation. The proposed framework is circuit independent and can be applied to any combinational circuit. Considering larger accelerator circuits, RETSINA can be directly used to produce their approximate reconfigurable counterparts. However, as shown in Fig. 13, the time complexity of RETSINA scales linearly with respect to the circuit's size. To avoid this overhead, RETSINA can be used to approximate the most intensive parts of the accelerator (e.g., its arithmetic units). Then a framework similar to [40], [41] can be used to build the entire approximate accelerator out of its approximate reconfigurable components produced by RETSINA.

Application Evaluation: Finally, we demonstrate the efficiency of the reconfigurable circuits generated by RETSINA at application level. We select Neural Network inference as our testcase and we use the ResNet-8 network [59] (7 convolution layers) trained with the CIFAR-10 image classification dataset. The NN is quantized at 8 bits to avoid floating point operations [60]. First, we use RETSINA to generate two approximate reconfigurable 8-bit multipliers. The first one, named RM1, is produced targeting three accuracy levels with error bounds {0%, 0.5%, 1.5%}. The second one, named RM2, is generated for two accuracy levels with error bounds {0.5%, 1.0%}. RM2 features only approximate levels and thus, the wires identified in the first accuracy level (i.e., 0.5%) are pruned instead of replaced with approximate switches. In our evaluation, as a reference point, we consider the Pareto-optimal fixed approximate multipliers (36 in total) of the state-of-the-art EvoApprox8b library [31]. For the fairness of the performed comparisons, the accurate 8-bit multiplier of [31] is used as the input circuit of RETSINA. All the circuits are synthesized and simulated at the critical path delay of the accurate 8-bit multiplier [31], i.e., we set zero delay overhead at RETSINA. RM1 achieves 8% and 17% energy reduction at the accuracy levels 2 and 3, respectively, while it induces 1% energy overhead at the accuracy level 1 (i.e., accurate mode). RM2 achieves 16% and 22% energy reduction at the accuracy levels 1 and 2, respectively, and does not support accurate execution. The accuracy levels of RM1 and RM2 are selected targeting high inference accuracy. To achieve this, we used [31] and [60] to profile how the multiplier's error impacts ResNet's accuracy. Next, we extend [60] and integrate our reconfigurable multipliers RM1 and RM2. In [60], approximate multipliers are used to perform the multiplications of each convolution layer. However, since different layers have different accuracy requirement, [60] employs an heterogeneous architecture that comprises several fixed approximate multiplier types. At runtime, at each layer, only one approximate multiplier type is used and the rest ones are power gated. This approach results in a significant area overhead and a potential performance/throughput loss due to the underutilized hardware. In our evaluation, we avoid these overheads and elucidate the significance our dynamically reconfigurable approximate circuits by considering an homogeneous architecture (i.e., only one approximate multiplier type is used either from [31] either RM1 or RM2). Since RM1 and RM2 feature several accuracy levels, we can dynamically select the desired accuracy level per layer at runtime. Therefore, for the ResNet-8, 2187 and 128 different configurations (i.e., selected accuracy level per layer) are obtained using RM1 and RM2, respectively. In Fig. 14, the accuracy-energy tradeoff of the ResNet-8 network is illustrated. To evaluate the energy savings, we measure the energy consumed in the multiplication operations (21.18M in total). The relative energy with respect to the energy consumed by the accurate multiplier is reported. The accuracy achieved using the accurate 8-bit multiplier is 83.26%. In Fig. 14, the multipliers of [31] and the configurations of RM1 and RM2 that feature up to 2% accuracy loss are depicted. Note that, to generate Fig. 14, all the configurations of RM1 and RM2 are evaluated and application mapping to the circuits generated by RETSINA is out of the scope of this paper. As shown, using approximate multipliers in NN inference can significantly decrease the energy consumption for a minimal accuracy loss. For only 0.5% accuracy loss, RM1 and RM2 achieve 15% and 19% energy reduction, respectively. In the respective

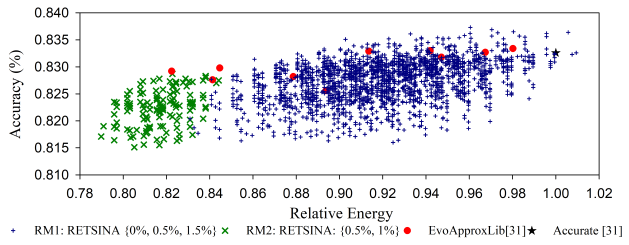


FIGURE 14. The accuracy-energy tradeoff of the quantized ResNet-8 when using approximate multipliers for the inference. The energy consumed in the multiplications is measured. Different red points are different multipliers of [31]. Different blue (green) points are different runtime configurations of the reconfigurable multiplier RM1 (RM2) produced by RETSINA.

configuration for RM1, accuracy level 3 was selected in 5 layers while accuracy levels 1 and 2 were selected in 1 layer each. For RM2, the accuracy level 1 was selected in 4 layers and the accuracy level 2 was selected in 3 layers. It is also observed that in many cases the ResNet's accuracy also increases (for both [31] and RM1/RM2). The latter is subject to the weight tuning performed by [60] to efficiently map NN inference on approximate hardware. The red points in Fig. 14 present the accuracy-energy tradeoff achieved when using fixed approximate multipliers [31]. However, different red points are produced by different approximate multiplier types. Thus, their efficiency over different NNs is unclear since, fixed approximation cannot guarantee that the quality requirements will be always met. It is worth mentioning that out of the 36 examined multipliers of [31], only 11 appear in Fig. 14 due to their high accuracy loss. On the other hand, the different blue (green) points are different runtime configurations obtained using RM1 (RM2). As a result, RM1 and RM2 enable the user to have a very fine-grained control of the applied approximation and the final accuracy. Thus, by dynamically selecting the accuracy level per layer, the energy savings can be maximized while the final accuracy loss is minimized. Fixed approximate multipliers [31] deliver higher energy saving since they avoid the cost of the accuracy reconfiguration circuitry. For example, for 0.5% MRED bound, [31] achieves 16% energy reduction. Nevertheless, as shown in Fig. 14, the runtime reconfiguration of RM1 and RM2 makes them more efficient at application level. Less sensitive layers can select higher approximation and the most sensitive ones can switch to more accurate execution. The accuracy-energy Pareto-front (Fig. 14) comprises mainly RM1 and RM2 configurations and only two fixed approximate multipliers [31] appear on the Pareto-front. Therefore, RM1 and RM2 not only enable layer significance-aware accuracy reconfiguration at runtime, but they also deliver very energy efficient solutions that outperform fixed approximation.

V. RELATED WORK

In this section related research in the field of approximate circuit design is discussed. Extensive research activity is reported in the design of approximate arithmetic circuits. A widely used approximation technique is logic simplification that mainly alters a cell's truth table. Exploiting this technique [4]–[6] produce approximate adders while [8]–[10] generate approximate multipliers and [15] designs approximate dividers. Another very efficient approximation technique is algorithmic approximation that modifies the implemented algorithm reducing its complexity. Algorithmic approximation is applied in [7] to obtain approximate adders, while [11]–[14] and [16]–[20] generate approximate multipliers and dividers, respectively. However, all these works produce architecture and operation dependent approximate circuits that feature only a single accuracy level. Moreover, their application as well as energy efficiency over differing architectures is not comprehensively analyzed.

Approximate computing increases the already complex task of hardware design by adding the error dimension in the systems design. To address the aforementioned inefficiency, significant research is reported in the field of approximate design automation. The authors in [25] and [26] exploit the "don't care" conditions to generate approximate logic circuits. In [27] a greedy approach is employed to detect and apply approximate transformations on the behavioral description of the circuit. Probabilistic Pruning and Logic Minimization [24] systematically remove components form a circuit and apply logic simplification based on the circuit's activity profile and the output significance. Approximate circuit synthesis by determining the error propagation and applying precision scaling is implemented in [28], while [29] further reduces the power consumption by leveraging the delay reduction to decrease the operating voltage value. Cartesian genetic programming is employed in [30], [31] to produce Pareto-optimal approximates multiplier and adders. The library's standard cells are approximated in [32] and a heuristic approach is devised to modify the accurate netlist using these cells and generate approximate arithmetic circuits. An extension to high-level synthesis tools [33] applies variable-to-constant approximation and builds approximate circuits that satisfy real-time constraints. The gate-level pruning framework, proposed in [21], replaces the circuit wires by constant values based on their output significance and their toggling count. The authors in [22] propose a circuit partition algorithm to quantify the output significance of every gate and use this algorithm to guide and accelerate the application of [21]. Multi-level approximation is applied in [23] where a voltage-driven gate-level pruning is used to maximize the application of voltage over-scaling. However, during the pruning procedure, [21]–[23] consider only the error value and do not examine the impact of pruning a wire on the circuit's power. Therefore, the power efficiency of the generated designs remains unclear. Extending [21], power, delay, and error estimators are used in [34] to assess the impact of pruning a node on the circuit's energy×delay. Nevertheless, only a 32-bit adder is examined in [34]. All these works produce static approximation circuits, neglecting the ability to dynamically reconfigure the accuracy at runtime.

Acknowledging the necessity to dynamically set the approximation level, considerable research interest is shown on reconfigurable approximate circuits. Reconfigurable approximate adders are produced in [42] by generating a dual state full adder that uses two multiplexers to select between the accurate and approximate results. An approximate carry look-ahead adder is designed in [43] that is able to switch between exact and approximate modes by applying power gating to switch off the exact part. Similarly, in [61], approximate reconfigurable multipliers are generated using dual quality 4:2 compressors that switch between accurate and approximate mode using power-gating. In [44], the addition is split to small accurate sub-adders and multiplexers select between the predicted approximate carry and the exact one. An iterative approximate reciprocal process is proposed in [46] to build a quality configurable approximate divider that delivers more accurate results as the number of iterations increases. Approximate exponential function unit is designed in [47] that achieves runtime accuracy reconfiguration by adjusting the number of added Taylor terms. In [45] the authors employ Cartesian genetic programming and clock gating to build approximate circuits that support approximate and accurate modes. Nevertheless, this technique is applied only on an 8-bit multiplier reporting significant overheads (power, area) for the exact mode, especially when targeting low error values. All these works examine only a specific circuit/operation type and/or support only one approximation level. Finally, an approximate Coarse-Grained Reconfigurable Arrays (CGRA) architecture is employed in [62]. The accurate multipliers and adders of the CGRA's processing elements are replaced by the dual-mode ones proposed in [43] and [61] to achieve runtime accuracy reconfiguration. Then, a mapping technique is proposed in [62] to map applications to this approximate CGRA. Therefore, RETSINA and [62] are co-operative by nature. RETSINA can be used to generate energy efficient approximate reconfigurable circuits and a framework similar to [62] can be used to enable efficient application mapping to these circuits. We differentiate from the state-of-the-art and propose, for the first time, an automated logic synthesis framework that generates dynamically reconfigurable approximate circuits that feature varying accuracy levels.

VI. CONCLUSION

Recently, approximate computing is established as a promising solution to design energy efficient circuits. However, the requirement for computational accuracy is not fixed at runtime and the impact of approximation to the end quality is highly input-dependent. In this work, we proposed and implemented an automated approximate design framework (RETSINA) to generate energy-efficient approximate circuits that feature dynamically reconfigurable accuracy at runtime and support several accuracy levels. Extensive experimental evaluation using industry-strength benchmarks and two state-of-the-art technologies demonstrates the efficiency of our framework. In addition, we examined the efficacy of our power-aware heuristic approach compared to the state-of-the-art GLP framework and showed that it better identifies the approximation wire candidates. Finally, through Neural Network inference application evaluation, we highlighted the high efficiency of the circuits generated by our framework in terms of both accuracy control as well as energy reduction.

REFERENCES

[1] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th Annu. Design Autom. Conf. (DAC)*, 2013, pp. 1–9.

[2] A. Yazdanbakhsh, D. Mahajan, P. Lotfi-Kamran, and H. Esmaeilzadeh, "Axbench: A benchmark suite for approximate computing across the system stack," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-CS-16-01, 2016.

[3] A. Raha and V. Raghunathan, "Towards full-system energy-accuracy tradeoffs: A case study of an approximate smart camera system," in *Proc. 54th Annu. Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[4] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," in *Proc. 13th IEEE Int. Conf. Nanotechnol. (IEEE-NANO)*, Aug. 2013, pp. 690–693.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[6] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2016, pp. 660–665.

[7] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed Truncation-Error-Tolerant adder and its application in digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[8] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.

[9] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.

[10] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[11] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 418–425.

[12] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBA multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.

[13] V. Leon, G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Walking through the energy-error Pareto frontier of approximate multipliers," *IEEE Micro*, vol. 38, no. 4, pp. 40–49, Jul. 2018.

[14] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[15] S. Venkatachalam, E. Adams, and S.-B. Ko, "Design of approximate restoring dividers," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[16] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the design of approximate restoring dividers for error-tolerant applications," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2522–2533, Aug. 2016.

[17] R. Zendegani, M. Kamal, A. Fayyazi, A. Afzali-Kusha, S. Safari, and M. Pedram, "SEERAD: A high speed yet energy-efficient rounding-based approximate divider," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2016, pp. 1481–1484.

[18] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram, and Z. Navabi, "TruncApp: A truncation-based approximate divider for energy efficient DSP applications," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1635–1638.

[19] H. Jiang, L. Liu, F. Lombardi, and J. Han, "Adaptive approximation in arithmetic circuits: A low-power unsigned divider design," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1411–1416.

[20] H. Jiang, L. Liu, F. Lombardi, and J. Han, "Low-power unsigned divider and square root circuit designs using adaptive approximation," *IEEE Trans. Comput.*, vol. 68, no. 11, pp. 1635–1646, Nov. 2019.

[21] J. Schlachter, V. Camus, K. V. Palem, and C. Enz, "Design and applications of approximate circuits by gate-level pruning," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1694–1702, May 2017.

[22] I. Scarabottolo, G. Ansaloni, G. A. Constantinides, and L. Pozzi, "Partition and propagate: An error derivation algorithm for the design of approximate circuits," in *Proc. 56th Annu. Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[23] G. Zervakis, K. Koliogeorgi, D. Anagnostos, N. Zompakis, and K. Siozios, "VADER: Voltage-driven netlist pruning for cross-layer approximate arithmetic circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 6, pp. 1460–1464, Jun. 2019.

[24] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, "Synthesizing parsimonious inexact circuits through probabilistic design techniques," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2s, pp. 93:1–93:26, May 2013.

[25] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, Jun. 2012, pp. 796–801.

[26] J. Miao, A. Gerstlauer, and M. Orshansky, "Multi-level approximate logic synthesis under general error constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 504–510.

[27] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "ABACUS: A technique for automated behavioral synthesis of approximate computing circuits," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2014, pp. 1–6.

[28] C. Li, W. Luo, S. S. Sapatnekar, and J. Hu, "Joint precision optimization and high level synthesis for approximate computing," in *Proc. 52nd Annu. Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[29] S. Lee, L. K. John, and A. Gerstlauer, "High-level synthesis of approximate hardware under joint precision and voltage scaling," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 187–192.

[30] M. Ceska, J. Matyas, V. Mrazek, L. Sekanina, Z. Vasicek, and T. Vojnar, "Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 416–423.

[31] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApproxSb: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 258–261.

[32] S. De, J. Huisken, and H. Corporaal, "An automated approximation methodology for arithmetic circuits," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[33] M. T. Leipnitz and G. L. Nazar, "High-level synthesis of approximate designs under real-time constraints," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5s, pp. 59:1–59:21, Oct. 2019.

[34] Z. Zhang, Y. He, J. He, X. Yi, Q. Li, and B. Zhang, "Optimal slope ranking: An approximate computing approach for circuit pruning," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–4.

[35] L. Chen, J. Han, W. Liu, and F. Lombardi, "Algorithm and design of a fully parallel approximate coordinate rotation digital computer (CORDIC)," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 3, no. 3, pp. 139–151, Jul. 2017.

[36] L. Chen, F. Lombardi, J. Han, and W. Liu, "A fully parallel approximate CORDIC design," in *Proc. Int. Symp. Nanosc. Archit.*, Jul. 2016, pp. 197–202.

[37] Q. Liao, W. Liu, F. Qiao, C. Wang, and F. Lombardi, "Design of approximate FFT with bit-width selection algorithms," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[38] W. Liu, Q. Liao, F. Qiao, W. Xia, C. Wang, and F. Lombardi, "Approximate designs for fast Fourier transform (FFT) with application to speech recognition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 12, pp. 4727–4739, Dec. 2019.

[39] G. Paim, L. M. G. Rocha, H. Amrouch, E. A. C. da Costa, S. Bampi, and J. Henkel, "A cross-layer gate-level-to-application co-simulation for design space exploration of approximate circuits in HEVC video encoders," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[40] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique, "autoAx: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components," in *Proc. 56th Annu. Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[41] G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Multi-level approximate accelerator synthesis under voltage island constraints," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 4, pp. 607–611, Apr. 2019.

[42] A. Raha, H. Jayakumar, and V. Raghunathan, "Input-based dynamic reconfiguration of approximate arithmetic units for video encoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 3, pp. 846–857, Mar. 2016.

[43] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "RAP-CLA: A reconfigurable approximate carry look-ahead adder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 8, pp. 1089–1093, Aug. 2018.

[44] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2013, pp. 48–54.

[45] V. Mrazek, Z. Vasicek, and L. Sekanina, "Design of quality-configurable approximate multipliers suitable for dynamic environment," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Aug. 2018, pp. 264–271.

[46] J. Melchert, S. Behroozi, J. Li, and Y. Kim, "SAADI-EC: A quality-configurable approximate divider for energy efficiency," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2680–2692, Nov. 2019.

[47] D. Wu, T. Chen, C. Chen, O. Ahia, J. S. Miguel, M. Lipasti, and Y. Kim, "SECO: A scalable accuracy approximate exponential function via cross-layer optimization," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[48] C. Piguet, "Low-power and low-voltage CMOS digital design," *Microelectron. Eng.*, vol. 39, nos. 1–4, pp. 179–208, Dec. 1997.

[49] Q. Xie, X. Lin, Y. Wang, S. Chen, M. J. Dousti, and M. Pedram, "Performance comparisons between 7-nm FinFET and conventional bulk CMOS standard cell libraries," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 8, pp. 761–765, Aug. 2015.

[50] L. Mencarelli, C. D'Ambrosio, A. Di Zio, and S. Martello, "Heuristics for the general multiple non-linear knapsack problem," *Electron. Notes Discrete Math.*, vol. 55, pp. 69–72, Nov. 2016.

[51] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[52] N. Weste and D. Harris, *Datapath Subsystems*, 4th ed. Reading, MA, USA: Addison-Wesley, 2010.

[53] J. Monteiro, S. Devadas, A. Ghosh, K. Keutzer, and J. White, "Estimation of average switching activity in combinational logic circuits using symbolic simulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 16, no. 1, pp. 121–127, Jan. 1997.

[54] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectron. J.*, vol. 53, pp. 105–115, Jul. 2016.

[55] H. Amrouch, S. Salamin, G. Pahwa, A. D. Gaidhane, J. Henkel, and Y. S. Chauhan, "Unveiling the impact of IR-drop on performance gain in NCFET-based processors," *IEEE Trans. Electron Devices*, vol. 66, no. 7, pp. 3215–3223, Jul. 2019.

[56] H. Amrouch, G. Pahwa, A. D. Gaidhane, J. Henkel, and Y. S. Chauhan, "Negative capacitance transistor to address the fundamental limitations in technology scaling: Processor performance," *IEEE Access*, vol. 6, pp. 52754–52765, 2018.

[57] M. Hoffmann, F. P. G. Fengler, M. Herzig, T. Mittmann, B. Max, U. Schroeder, R. Negrea, P. Lucian, S. Slesazeck, and T. Mikolajick, "Unveiling the double-well energy landscape in a ferroelectric layer," *Nature*, vol. 565, no. 7740, pp. 464–467, Jan. 2019.

[58] J. C. Wong and S. Salahuddin, "Negative capacitance transistors," *Proc. IEEE*, vol. 107, no. 1, pp. 49–62, Jan. 2019.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, pp. 770–778, Dec. 2015.

[60] V. Mrazek, Z. Vasicek, L. Sekanina, M. A. Hanif, and M. Shafique, "ALWANN: Automatic layer-wise approximation of deep neural network accelerators without retraining," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[61] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1352–1361, Apr. 2017.

[62] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, and M. Shafique, "X-CGRA: An energy-efficient approximate coarse-grained reconfigurable architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, to be published.

GEORGIOS ZERVAKIS received the Diploma and Ph.D. degrees from the Department of Electrical and Computer Engineering (ECE), National Technical University of Athens (NTUA), Greece, in 2012 and 2018, respectively. He is currently a Postdoctoral Researcher with the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany. Before joining KIT, he worked as a Primary Researcher in several EU-funded projects as a member the Institute of Communication and Computer Systems (ICCS). His research interests include approximate computing, low-power design, design automation, and integration of hardware acceleration in cloud.

HUSSAM AMROUCH (Member, IEEE) received the Ph.D. degree *(summa cum laude)* from KIT, in 2015. He is currently a Research Group Leader with the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany. He is also leading the Dependable Hardware Research Group. His main research interests are emerging technologies, design for reliability from physics to system level, and machine learning. He holds seven HiPEAC Paper Awards. He has three Best Paper Nominations at DAC'16, DAC'17, and DATE'17 for his work on reliability. He serves as an Associate Editor for *Integration* and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) Journal.

JÖRG HENKEL (Fellow, IEEE) received the Diploma and Ph.D. *(summa cum laude)* degrees from the Technical University of Braunschweig. He is currently with the Chair Professor for Embedded Systems, Karlsruhe Institute of Technology. Before that he was a Research Staff Member at the NEC Laboratories, Princeton, NJ, USA. His research work is focused on co-design for embedded hardware/software systems with respect to power, thermal, and reliability aspects. He has received six best paper awards throughout his career from, among others, ICCAD, ESWeek, and DATE. For two consecutive terms he has served as the Editor-in-Chief of the *ACM Transactions on Embedded Computing Systems*. He is also the Editor-in-Chief of *IEEE Design & Test Computers* Magazine and is/has been an Associate Editor for major ACM and IEEE journals. He has led several conferences as the General Chair incl. ICCAD and ESWeek and serves as a steering committee chair/member for leading conferences and journals for embedded and cyber-physical systems. He coordinates the DFG Program SPP 1500 Dependable Embedded Systems. He is also a Site Coordinator of the DFG TR89 Collaborative Research Center on Invasive Computing. He is also the Chairman of the IEEE Computer Society and the Germany Chapter.